WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Friday, January 21, 2005

Hide?	Set Name	Query	Hit Count
	DB=USPT; PLUR=YES; OP=ADJ		
☐	L13	modification.ti. and web.ti. and page$.ti.	3
☐	L12	modif$4 same (web adj2 page$) and applet$ and (html or hyper text markup language) and (tag near identifier$)	7
☐	L11	L8 and (html or hyper text markup language)	10
☐	L10	L8 and (http or hyper text transfer protocol)	10
☐	L9	L8 and (html or hyper text transfer protocol)	10
☐	L8	L1 and script$ and tcp and (ip or internet protocol) and (exchang$4 same information)	10
☐	L7	L6 and script and tcp and (ip or internet protocol)	1
☐	L6	L1 and 709/2$$.ccls. and (Web adj2 page$) same modification$	4
☐	L5	L1 and 709/2$$.ccls.	27
☐	L4	L1 and 709/2$4.ccls.	27
☐	L3	L2 and (tag near identifier$)	0
☐	L2	L1 and script and tcp and (ip or internet protocol)	14
☐	L1	modify$4 same (web adj2 page$) same applet$	45

END OF SEARCH HISTORY

h e b b cg b chh e e f ff e ch e h

L6: Entry 1 of 4 File: USPT Jul 20, 2004

DOCUMENT-IDENTIFIER: US 6766356 B1
TITLE: Method and system for remotely modifying presentations in a multimedia conference

Brief Summary Text (4):
Use of internet-related networks (e.g. the World Wide Web) for multi-media conferencing presentations has increased dramatically in recent years. A conference participant may direct another participant to information stored on a computer network (e.g. web pages). U.S. Pat. No. 5,944,791 describes an example of a presentation using the web. To facilitate the conference it is desirable for a conference participant (e.g., a conference moderator) to be able to initiate and direct other participants' receipt of such information (so-called collaborative web browsing). To do this, the moderator must send command data, such as uniform resource locator (URL) data to other participants' digital processing system (DPS) to direct the other participants to a particular web page. The web page may contain embedded Javascript that causes the display of the web page to be modified in a predetermined way. For example, the embedded Javascript could contain a timeline that allows the conference participants to view some of the information on the web page and after some time view additional or different information of the same web page. The Javascript is downloaded with the hypertext markup language (HTML), not necessarily embedded with it. A library of Javascript is downloaded to the participant DPS's cache memory. The moderator can then access the Javascript necessary to cause a predefined desired action, and cause that Javascript to be embedded into a web page which a participant is viewing. FIG. 1 shows a method by which the moderator causes the modification of a displayed web page of a participant. The method 100 of FIG. 1 begins with operation 105, in which a moderator DPS in a web conference directs a participant to a web page. In operation 110 the participant is directed to a web page and receives a library of Javascript along with the HTML. In operation 115 the moderator issues commands that invoke the Javascript received at the participant's DPS. This Javascript causes a predefined action such as a further portion of the display becoming visible or an icon appearing. The Javascript is considered static in that once the HTML and Javascript are downloaded for a given web page the associated library of Javascript cannot be modified. The moderator can invoke selected Javascript from the library to cause desired actions on the display of the participant, but the moderator cannot change the library of available Javascript. The moderator of a web conference may direct a participant to a web page and then wish to modify what the participant is viewing beyond the Javascript transmitted with the HTML page. Currently, one way to do this would be to retransmit the web page with different embedded Javascript (i.e. a new web page). This method is depicted in FIG. 2. The method 200 depicted in FIG. 2 begins at operation 205 in which the moderator of a web conference directs a participant of the same web conference to a web page. In operation 210 the participant receives the static Javascript from the web page. This static Javascript is invoked by the moderator in operation 215 to cause predefined actions at the participant's DPS. This process would have a detrimental affect on the web conference because transmitting the new page would cause delay and a disruption of visual continuity as the participant's display screen would blank out while the new page was loaded.

Brief Summary Text (5):

h e b b g e e e f c e e e ge

Other methods of effecting a modification of a web page (e.g. mouse rollover or web page annotation overlay) are not part of an overall web conferencing scheme. The annotation overlay method of modifying a web page is depicted in FIG. 3. The method 300 begins at operation 305 in which a first web page 306 is received at a DPS. In operation 310 while the first web page 306 is being presented, a second web page 311 is caused to be received. The second web page 311 could be caused to be received by the transmitter of the first web page, by the receiver of the first web page, or by another entity. In operation 315 the second web page 311 lays over the first web page 306 and the presentation of the first web page 306 appears as modified web page 316 (i.e. the visual effect is that of modifying the first web page 306). The web page annotation over lay method depicted in FIG. 3 is analogous to placing a Post-it note on a document. In both cases the document itself is not changed, only its presentation.

Detailed Description Text (11):
In operation 715, the participant receives the URL and the Javascript for the web page and loads the web page and runs the Javascript. In operation 720, the participant receives the commands, which may cause the Javascript and HTML at the participant's DPS to be created or modified without reloading the web page. In operation 725, the moderator transmits further modifying commands to the participant's DPS, which may cause further modification of the Javascript and the HTNL in the same web page.

Detailed Description Text (13):
The set of available annotation containers 815 are implemented on each presentation page at the lowest level. In Appendix A, section 1 contains an example of the code for implementing annotations, section 2 contains an example of code which allocates the annotations containers on an HTML page, and section 3 contains an example of code that picks one of the annotation containers for use. The implementation of these containers depends on the specific browser (e.g. for Netscapes's Navigator the container is based on a LAYER tag and for Microsoft's Internet Explorer the container is based on a DIV tag). Within these container tags, HTML and Javascript code is inserted as defined by commands originated by the moderator. This contained HTML code can be dynamically and arbitrarily changed by a command initiated by the conference moderator. Appendix B contains an example of the code for changing the HTML within annotation container. The containers 815 can contain many types of annotation commands, which modify the presentation in various ways. For example, in an exemplary embodiment, the containers may contain a highlighting command to emphasize some portion of the presentation, a moving command to move portions of the presentation within the web page, and a sizing command. Appendix C contains an example of the code for changing an annotation's visibility, position, and size. The containers may contain other information or commands, for example a command to add or delete text, a Java applet, or web pages.

Detailed Description Text (17):
Thus, in one embodiment of the present invention, it will be possible for multimedia conference participant to be directed to a particular web page and while the information is presented to remotely receive commands, which can modify the existing Javascript, or HTML, or cause new Javascript or HTML to be created. The modifications to the presentation are not predetermined, but can be changed by the moderator at any time. In an alternative embodiment the modification commands are not issued by the moderator, but are issued by a presenter who may be another remote conference participant.

Current US Original Classification (1):
709/204

Current US Cross Reference Classification (1):
709/205

h e b b g e e e f c e e e ge

Current US Cross Reference Classification (2):
709/230

Previous Doc Next Doc Go to Doc#

L7: Entry 1 of 1 File: USPT Jul 4, 2000

DOCUMENT-IDENTIFIER: US 6085224 A
TITLE: Method and system for responding to hidden data and programs in a datastream

Abstract Text (1):
A system and method for detecting trigger events in datastreams received over a computer communication network is disclosed. The system includes an interceptor for intercepting datastreams from a computer network intended for an application program; a scanner for scanning the intercepted datastream for trigger events, such as cookie data, script commands, and applet programs; and an event response generator for processing detected trigger events. Configuration data is used to identify a response for trigger events such as disabling script commands or programs and deleting or modifying cookie data. The event indicators and an action menu are generated by the event response generator and delivered with the processed datastream to the application program. The application program displays the event indicators so the user is made aware of the trigger events and the action menu allows a user to respond to the detected trigger events. In the preferred implementation, the user may respond by obtaining information about the site which transmitted the datastream having the trigger events and then send e-mail to the administrator of the site. Other actions include modifying the configuration data so subsequent datastreams with the trigger event is passed by the system. Outbound messages from the application program are also intercepted and scanned for trigger events. In the preferred implementation, the configuration data are exchanged between the system and the application program so the user may modify the operation of the system. The configuration data are deleted from the outbound datastream before it is transmitted in the preferred implementation. The system and method of the present invention allow a user to view detected trigger events which otherwise would occur without the user's knowledge and provides the user with sufficient information so the user can make an informed decision as to whether to accept trigger events in a datastream from another site.

Brief Summary Text (5):
The communication stack which processes data messages for a computer is a group of programs that correspond to communication layers which are executed sequentially in a manner which visually resemble a stack. On the Internet, the communication stack implements a Transport Control Protocol/Internet Protocol (TCP/IP). The implementation of this protocol is usually perceived to include at least three layers of processing. The first layer is a data link layer which maps an address from the hardware component to an Internet address for messages inbound to the computer and maps an Internet address to a hardware address for outbound messages. The next layer in the stack is the network layer which verifies that the network protocol parameters for the data communication are correct. Finally, the transport layer verifies that the datastream portion of the communication has been properly delivered and makes the datastream available for an application program. Datastream as used herein refers to

Brief Summary Text (8):
In a response to a request for a page sent by a browser, a server sends multiple HTML files which comprise the page in messages implemented in the HTTP protocol.

h e b b g e e e f c e e e ge

When the HTML file or files are received by the computer executing the browser, each communication stack layer performs its function until a datastream containing an HTTP header and corresponding data segment is presented to the browser. One portion of the browser verifies that the information and the HTTP header have been accurately delivered to the application program. The browser then displays the data delivered in the HTML files received from the server. Because the TCP/IP protocol used for the Internet is a packet communication protocol, several messages are probably required before a complete file is available for display. Besides graphical data, the HTML file also contains data and/or commands which may not be displayed at the browser. This "hidden" data and/or commands may be used to cause the computer executing the browser to store information or execute programs without the user's knowledge of the existence or purpose of the information or program.

Brief Summary Text (11):
Recently, powerful interpretive languages have been developed which may be executed in a browser. Known interpretive languages are JAVA developed by Sun Microsystems, Javascript developed by Netscape Communications Corporation, and Visual Basic Script developed by Microsoft Corporation. Because each one of these languages are interpreted, a program written in one of these languages does not need to be compiled with prior knowledge of the computer on which it will execute. Instead, the interpreter executes within the application space for the application program, such as the browser, and this interpreter executes statements received in a file containing the interpretative language statements. Files containing interpretive language statements are known as applets. While applets have a number of beneficial purposes, they may also cause problems. For example, a JAVA applet may be imbedded in an HTML file, sent to a user's computer and executed by an interpreter in the browser without the user's knowledge. Such programs may be used to gain unauthorized access to resources or data on the user's computer. Additionally, these interpretive language programs may include cookie commands that identify tracking data as discussed above. These cookie commands are part of the data segment of a datastream for a browser and not part of the HTTP header. As a result, these cookie commands are not detected by the programs that may be used to detect and delete cookie data from HTTP headers.

Brief Summary Text (19):
In the method of the present invention, an inbound datastream is intercepted prior to it being delivered to an application program. The datastream is then scanned for trigger events and processed according to the configuration data for the server site. If the configuration data indicates that script programs are to be disabled, for example, any script program in the datastream is disabled so it cannot be executed by the browser. The detected trigger events are then logged and a data envelope containing event indicators and an action menu is generated and coupled to the processed datastream. The browser program then displays the processed datastream, the event indicators and the action menu. The user may select actions in the action menu to view detected trigger events or initiate control actions. For example, a user may view a disabled script program and conclude its execution is acceptable to the user. The user may then modify the configuration data to allow script program execution and then cause the browser to request the page from the server again. Upon receipt of the requested page, the system of the present invention does not disable the script program and it will execute when the datastream is passed to the browser.

Detailed Description Text (9):
In the preferred implementation, the most significant digit in the preferred 10 digit action map data defines whether hidden data from a server site is displayed in the browser. In response to this digit having a value of `0`, event response generator 34 deletes cookie values in an HTTP header or in embedded script "document.cookie" commands of a datastream received from the server site, sets the cache value in the HTTP header so data is not stored in the cache of the user's computer, and deletes at browser initiation or termination cookie data in the

"cookies.txt" file which were received from unauthorized servers. In response to this digit having a value `1`, event response generator 34 deletes the expiration dates for cookie data in HTTP header so the cookie values from the server site may be received, sets the cache value in the HTTP header so data is not stored in the cache of the user's computer, and deletes at browser initiation or termination cookie data in the "cookies.txt" file which were received from unauthorized servers. The second digit of the preferred action map data defines whether data from a server is stored on the user's disk drive. In response to this digit having a value of `0`, event response generator 34 deletes cookie values in an HTTP header for a datastream received from the server site, sets the cache value in the HTTP header so data is not stored in the cache of the user's computer, and deletes at browser initiation or termination cookie data in the "cookies.txt" file which were received from unauthorized servers. In response to this digit having a value `1`, event response generator 34 allows cookie values with valid expiration dates to be stored in the "cookies.txt" file and datastreams from a server may be stored in the cache of the user's computer and on the user's disk drive. The third digit of the preferred action map data defines whether data from a server stored on a user's disk drive may be returned to the server. In response to this digit having a value of `0`, event response generator 34 deletes all cookie values in an HTTP header for outgoing datastreams to the server site. In response to this digit having a value `1`, event response generator 34 allows cookie values to be returned to the server site which sent them in a previous datastream. The fourth digit of the preferred action map data defines whether a server can request data from another site for display by a user's browser. In response to this digit having a value of `0`, event response generator 34 deletes all HTML "<IMG src=`URL`>" statements where the URL identifies a server other than the one with which communication is active. In response to this digit having a value `1`, event response generator 34 allows GET requests for URLs from other server sites to be sent to the user's computer for display. The fifth digit of the preferred action map data defines whether a server can request data from another site be sent to the user's browser for storage. In response to this digit having a value of `0`, event response generator 34 deletes all HTML "Set Cookie" statements where the URL in the datastream identifies a server other than the one with which communication is active and no cache storage is allowed for datastreams having a server address different from the one with which communication is currently active. In response to this digit having a value `1`, event response generator 34 allows cookies from other server sites to be stored on the user's computer.

Detailed Description Text (10):
Continuing with the description of the preferred implementation, the sixth digit in the preferred 10 digit action map data defines whether programs from a server may be executed in the display of a user's browser. In response to this digit having a value of `0`, event response generator 34 disables HTML SCRIPT and APPLET tags as well as Javascript "document.applet" commands. In response to this digit having a value `1`, event response generator 34 allows HTML SCRIPT and APPLET tags as well as Javascript applets to execute in the display of a user's browser. The seventh digit of the preferred action map data defines whether programs from a server may execute on a user's computer. In response to this digit having a value of `0`, event response generator 34 disables HTML OBJECT and EMBED tags as well as Javascript "document.embed" commands. In response to this digit having a value `1`, event response generator 34 allows HTML OBJECT and EMBED tags as well as Javascript embedded commands to execute on the user's computer. The eighth digit of the preferred action map data defines whether browser and e-mail user information data may be sent to a server. In response to this digit having a value of `0`, event response generator 34 deletes "User-Agent" and "From:" fields from HTTP headers of outbound datastreams. In response to this digit having a value `1`, event response generator 34 allows HTTP headers having "User-Agent" and "From:" fields to be transmitted in outbound datastreams. The ninth digit of the preferred action map data defines whether page updates from a server may be received by a user's browser. In response to

Detailed Description Text (11):
this digit having a value of `0`, event response generator 34 deletes all "Connection:keep-alive" and "refresh" statements from HTTP headers in inbound datastreams. In response to this digit having a value `1`, event response generator 34 allows "Connection:keep-alive" and "refresh" statements to remain in HTTP headers in inbound datastreams so they are processed by a user's browser. The tenth digit of the preferred action map data defines whether disk I/O is active during network communication. In response to this digit having a value of `0`, event response generator 34 allows disk I/O to remain active during an active TCP/IP socket connection. In response to this digit having a value `1`, event response generator 34 disables disk I/O whenever a TCP/IP socket connection is active. Although these are the preferred actions and their preferred implementations in the present invention, other actions and implementations may be used without departing from the principles of the present invention.

Detailed Description Text (13):
Thus, the event configuration data defines the actions to be performed for each type of trigger event. When event response generator receives an action map from the user's browser, the event configuration data is modified to conform to the action map. In this manner, the user need only specify actions and the inventive system correlates the specified actions to the trigger events for which scanner 32 scans. In the preferred implementation, the most significant digit in the preferred 12 digit map data defines whether cookie values in an HTTP header for a data message received from another computer are modified or stored in a "cookies.txt" file. As shown in the table, a zero value for the digit indicates that no modifications are made to cookie values in the header, a "1" value for the digit indicates that a user may modify the value before it is returned to the computer which sent the message with the cookie data in the header, and the value "2" causes the file ("cookies.txt") in which cookie data are normally stored to be deleted upon initiation of the application program or upon termination of the application program. For the second map data digit, the value zero permits refresh files to be received and displayed by the application program and the value "1" deletes refresh file requests from outbound datastreams. Refresh files are typically HTML files sent by a server to update an area within a previously transmitted page. The third map data digit determines whether repeat images from the server are displayed by the application program. A value of zero permits the repeat data files to be received and displayed, a value of "1" preferably causes the TCP/IP socket to close after a Web page has been downloaded so repeat data files are not received. The fourth map digit defines whether MIME encoded files are decoded and displayed by the application program. A value of zero permits all MIME encoded files to be decoded and used by the application file, a value of "1" permits those MIME files containing text and image data only to be received and decoded by the application program, and the value "2" permits those MIME files containing text only to be received and decoded. The fifth map data digit either allows or disables execution of script commands received in a datastream. A value of zero for this digit permits the application program to receive and execute script commands while a value of "1" causes the scanner to disable script commands. Preferably, script commands are disabled by placing a comment character before the script command so the detected script command may be displayed by the application program for the user. In the preferred implementation, detected script commands are in Javascript or Visual Basic Script (VBS) languages. The sixth map data digit determines whether cookie data in script commands are permitted. That is, cookie data may be defined in script commands to avoid being detected in HTTP headers. If this map digit is zero, the script cookie data are received and used by the application program. If the map data value is "1", detected cookie data in script commands are disabled and displayed by the application program for the user so the user may determine whether receipt of the cookie data in the script command is allowed. The seventh digit in the map data determines whether a FORM "submit" script command will be provided to the application program. A digit value of zero allows FORM "submit" script commands

to be received and executed by the application program and the value of one causes the scanner to disable the FORM "submit" command so it may be displayed for the user by the application program without execution. The eighth digit in the map data determines whether a script program may execute a plug-in program using an "embed" command. A value of zero permits script programs to execute plug-in programs with "embed" commands and a value of "1" causes event response generator 34 to disable "embed" commands for plug-in programs in script programs. Preferably, script commands are disabled by either placing a comment character in front of the command which invokes the program or by inserting a return statement as the first statement in the imbedded program function so the body of the imbedded program function is not executed. The ninth digit of the map data determines whether an applet program is received and executed. When this digit is a zero, applet programs may be received and executed by the application program and when this digit is a "1", event response generator 34 disables the applet program. Again, the preferred implementation disables the applet program by either placing a comment character before the command invoking the program or placing a return statement as the first statement in the applet program. The tenth digit in the map data determines whether imbedded programs for a plug-in application are received and executed by the application program. A value of zero for this digit permits imbedded programs to be received and executed by the plug-in application and a value of "1" disables the imbedded programs for the plug-in application. Again, the preferred implementation disables imbedded programs or plug-in applications in the manner discussed with respect to applet programs. The eleventh map data digit determines whether data objects are passed and used by the application program. The map value of zero permits data objects to be received and used by the application program and a value of "1" disables data objects so a detected object may be displayed by the application program for the user. The data object is preferably disabled by placing a comment character in front of the command which uses the data object or by detecting the program portion component of the data object and disabling it by putting a return statement as the first statement in the program portion of the object. In the preferred implementation, data objects are typically data objects written in the Active X language. The twelfth digit of the map data determines whether disk I/O is disabled. When the value of this digit is at zero, disk I/O is disabled and files and data are written to a RAM area. When this digit is a one, files and data may be stored on a disk drive.

Detailed Description Text (21):
Examples of embedded commands include any commands which activate or execute a program or applet. Programs which may be activated by an embedded command include those written in the JAVA script or Visual Basic Script languages. Also, JAVA applets, Navigator plug in applications, and Microsoft Active-X control applications are programs that may be activated by embedded commands. For example, the HTML tag "APPLET" may be used to invoke a JAVA applet, the "EMBED" HTML tag may be used to invoke Netscape plug-in of applications, and the "SCRIPT" HTML tag may be used to invoke Javascript or Visual Basic Script programs. Thus, identification of these HTML tags in a file received from a server site is a detection of a trigger event for an embedded command. Also, Javascript programs received from a server site are scanned to determine if a "document.applets" string or "document.embeds" string is contained in the Javascript program. These two examples of JAVA script language statements are used to invoke a JAVA applet or a Netscape plug-in application, respectively. Again, these commands which activate programs or applets are merely examples of the types of commands which may be detected by scanner 32 in a scan of an HTML file or downloaded program or applet file.

Detailed Description Text (26):
The datastream and coupled data envelope are then displayed by the application program and the user may view the detected events. A user may then select an action in the action menu. In the preferred implementation, the actions in the action menu are performed by a script program invoked from the action menu. The actions available to a user in the preferred implementation include obtaining more

information about the server site which sent the detected trigger event or modification of the action map data so a trigger event may be used or executed. For example, to obtain more information about the server site, the user may send a WHOIS query to the domain name registration authority. This query is sent in the preferred embodiment by establishing a HTTP session, or alternatively a TELNET communications session, for transmission of the query to the domain name registration authority. The response to the query identifies the owner of the server site, its administrator, company name, and e-mail address. The user may now activate a selection in the action menu to send the identified administrator an e-mail regarding the detected trigger event. In the preferred implementation, a default message is provided for transmission to the administrator. In another aspect of the preferred implementation, the user may activate a selection in the action menu to send a FINGER query to the server site to obtain information regarding the administrator of the site. This information may also be used to send an e-mail to the administrator regarding the detected trigger event. After viewing the detected trigger event, a user may determine that the detected trigger event is acceptable to the user. In this case, the user selects an action from the action menu which modifies a value in the action map data so that subsequent transmissions permit the trigger event to be received and used or executed.

Detailed Description Paragraph Table (2):

DATA DIGIT POSITION	RESTRICTION VALUES	MAP 1
cookie.sub.-- modifications `0` = no modifications `1` = modify name = value statement or expiration date for cookie value `2` = delete cookies.txt at start-up/shut down 2 refresh, `0` = allow always `1` = delete always 3 keep.sub.-- alive, `0` = allow always `1` = delete always 4 mime.sub.-- type, `0` = allow all `1` = text and image only `2` = text only 5 script, `0` = allow always (Java Script or VBS) `1` = disable always 6 script.sub.-- cookie, `0` = allow always `1` = disable always 7 script.sub.-- submit, `0` = allow always `1` = disable always 8 script.sub.-- embed, `0` = allow always `1` = disable always 9 applet, `0` = allow always (Java) `1` = disable always 10 embed, `0` = allow always (Plug In) `1` = disable always 11 object; `0` = allow always (Active X) `1` = disable always 12 Virtual RAM Drive `0` = disable `1` = enable		

Current US Original Classification (1):
709/203

Current US Cross Reference Classification (1):
709/219

CLAIMS:

5. The system of claim 4 further comprising:

a script language program for implementing said action menu.

Previous Doc Next Doc Go to Doc#